Resumable Uploads

draft-ietf-httpbis-resumable-upload

IETF 124: November 5, 2025

Marius Kleidl, Lucas Pardue, Guoye Zhang

What changed since -08?

- WGLC, detailed feedback from Mike Bishop, Glenn Strauss, Willy Tarreau.
- Require Accept-Patch in OPTIONS response.
- Add security consideration regarding time-of-check to time-of-use bugs.
- Relax requirement on Upload-Complete in final responses.
- Relax requirements on limit changes.
- Describe the interaction between 100 and 104 responses.
- Numerous editorial improvements.

Current draft status

- ~~No open issues/pull requests 🎉~~
- 28 open issues (10 editorial, 4 need discussions)
- 6 open pull requests

Retrieving lost response after completed upload (#3270)

- If the client does not receive the final response after completing the upload, it cannot recover said response and upload was useless
- Workaround is to retrieve some information via HEAD requests
- Known pain point in existing deployments

Proposal:

- Server caches final response as long as upload resource exists
- Client can re-fetch final response using a 0-length PATCH request
  - No need for “appending to complete upload” error
  - No special client logic for re-fetching response necessary
  - Uploads become similar to state synchronization
  - There seems to be good support on GitHub for this

GET requests to upload resource (#3214)

- RFC 9110 tightly couples HEAD and GET requests
  - “The server SHOULD send the same header fields in response to a HEAD request as it would have sent if the request method had been GET” [RFC 9110]
- Our draft describes how a HEAD response should look like
  - What about GET requests?
- We don’t want to get into downloads!

Proposal:

- Upload resource responds with 2XX to GET requests
- Include Upload-* headers similar to HEAD responses
- No requirements on response content

Forward offset jumps (#3317)

- Offset = number of bytes received by the upload resource
- Can the offset jump forward without bytes being transferred?
  - E.g. server fills in representation data partially from other source (e.g. previous upload)
- Client would skip uploading these bytes and then continue
  - Current draft allows this behavior, but it’s not mandatory
  - Allows recovering from previous upload when the client didn’t learn upload resource URL

Question:

- Should clients be required to follow an offset that is greater than the amount of bytes it uploaded (e.g. because another client uploaded the additional bytes)?

Request size granularity (#3193)

- Implementing resumable uploads on legacy systems sometimes requires adhering to storage granularities
  - I.e., data can only be appended in multiples of a specific granularity (e.g. 50 MiB)
- Draft has implicit support for this:
  - Server can discard trailing bytes to meet granularity and responds with 409
  - Client resumes upload and resend discarded bytes

Questions:

- Should the draft contain explicit considerations to handle granularities?
- Should there be specific limit values to indicate a required granularity?

Thank you!